

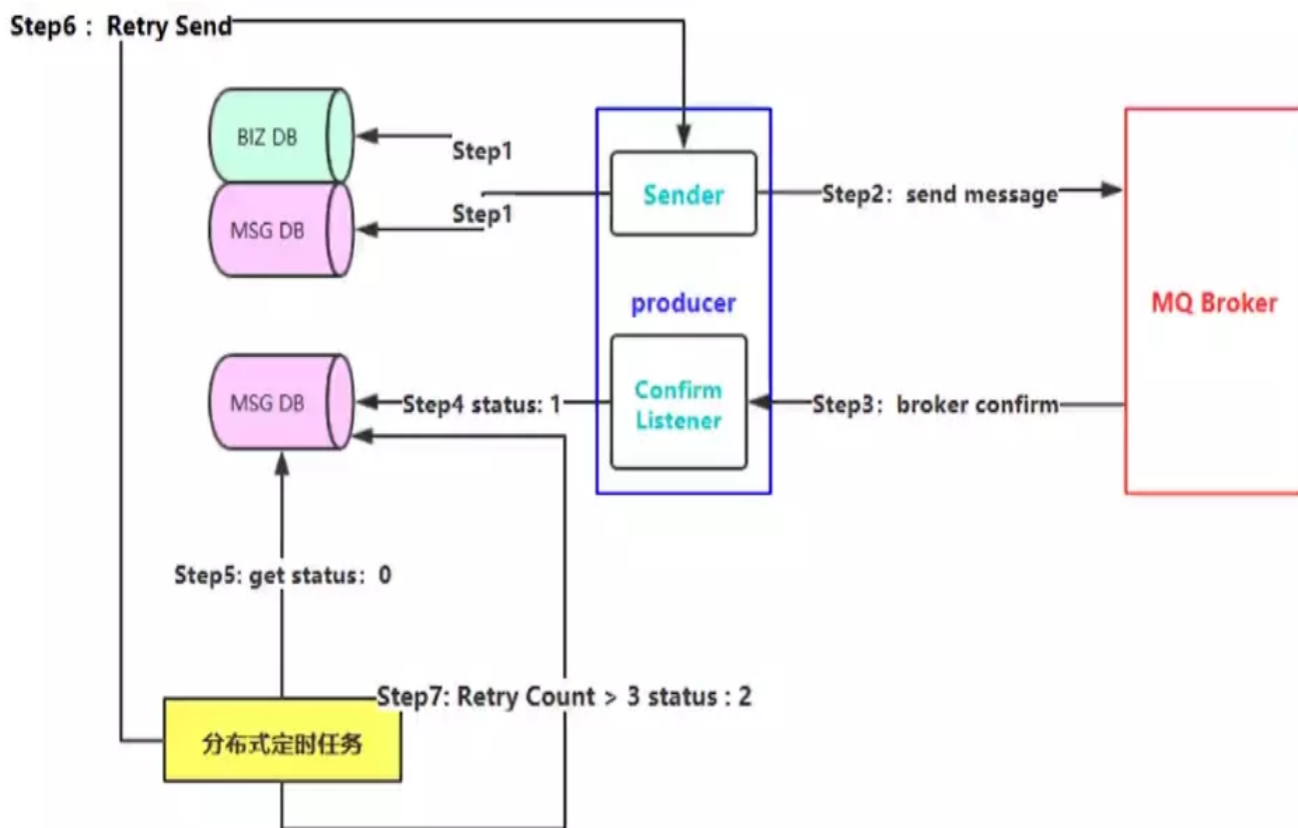
3.1) 如何保障消息的可靠性投递

一:如何保障消息的百分百投递成功

1.1)什么是生产端的可靠性投递

- ①:保障消息成功发送出去
- ②:保障mq节点成功接收消息
- ③:消息发送端需要收到mq服务的确认应答
- ④:完善的消息补偿机制（百分百成功成功 需要该步骤）

1.2) 解决保障可靠性投递的方案（消息落库）



消息入库打标解决思路（Order_Serve 调用物流服务举例）

在消息生产者端(也就是订单服务)

正常链路流程

第一步(该环节调用了操作了二次数据库):在创建订单的操作的时候,把数据插入到订单相关的表中,并且构造调用物流模块的数据消息,把消息插入到消息表中,初始状态为0

第二步:把物流消息投递到消息队列中,

第三步:消息队列访问一个确认消息,并且由,订单服务来监控mq server的确认消息

第四步:根据收到的确认消息来更新数据库中的消息记录的状态

异常链路流程

第一步(该环节调用了操作了二次数据库):在创建订单的操作的时候,把数据插入到订单相关的表中,并且构造调用物流模块的数据消息,把消息插入到消息表中,初始状态为0

第二步:把物流消息投递到消息队列中,

第三步:由于网络闪断,导致消费端监控mq服务访问的确认消息没有收到,那么在msg_db中的那条消息的状态永远就是0状态。这个时候,我们需要对这种情况下做出补偿

补偿机制:

启动一个分布式的定时任务,不定时的去扫描msg_db的这个表,状态为0的消息记录,在这里我们可以根据业务来设置扫描重发规则

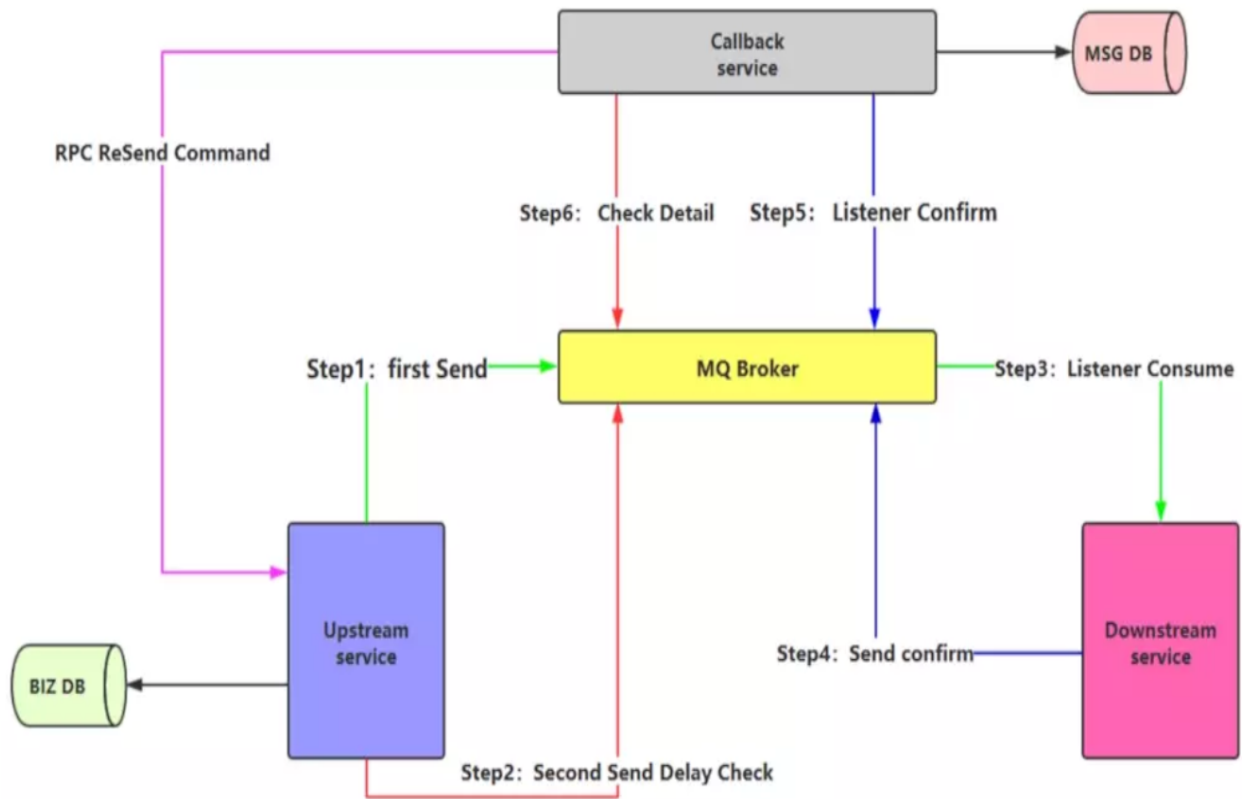
规则1:插入msg_db表中5Min后状态还是为0的记录,进行消息重试

规则2:若重试的次数超过五次状态还是为0的话,我们就把消息状态改为2,此时我们需要人工的去确认状态为2的消息是什么原因导致没有成功的

消息入库打标的缺点:

在第一步的过程中,既插入了业务数据表,也同时插入了消息记录表,进行了二次db操作,在高并发的环境下,这个环境就会造成性能瓶颈

1.3) 延时投递,做二次确认检测,回调检测



还是由order服务(upstream service) 物流服务 (downstream servcie) 来举例